Introduction to Core Data on iPhone

Session 421

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Introduction to Core Data on iPhone

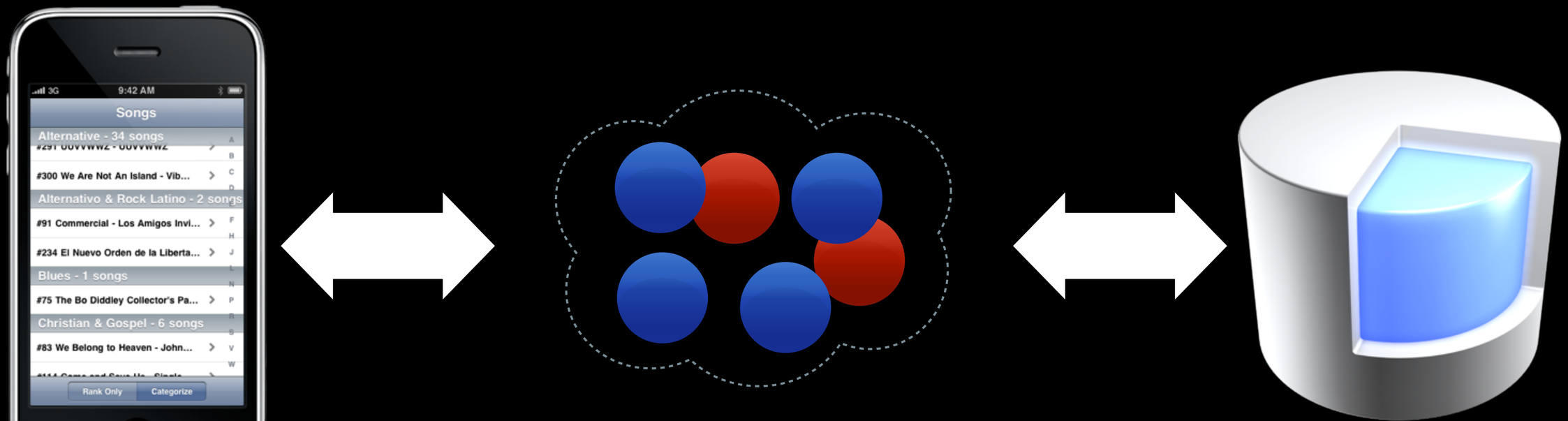
Miguel Sanchez
Core Data Engineering Team

Introduction

- Core Data is a powerful and efficient framework for data management and persistence
- Newly available in iPhone OS 3.0
- On Mac OS X since 10.4 (Tiger)

How Can Core Data Help You?

Management of application data



- Read/save objects from file
- Maintain graph of objects
- Efficient speed and memory

Agenda

Introduction to Core Data technology

- Modeling your data
- Working with objects
- Displaying results

Tomorrow

What's New In Core Data	North Beach Wednesday 10:30AM

Core Data Framework Overview

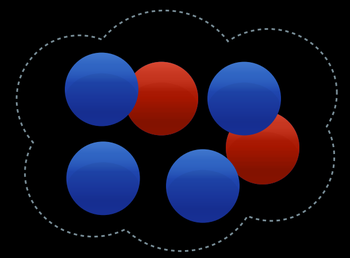
Core Data Benefits

Think objects; avoid database grunt work

- Efficient fetching and saving
- Change tracking and undo
- Object validation and relationship maintenance
- Performance

How Do We Do This?

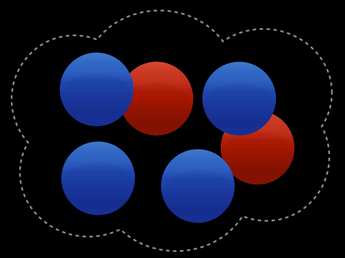
We need a model



Name	Ben
Name	Miguel
Name	Adam
Name	Melissa

Introducing Our Vocabulary

Common terms



Managed Objects = Data Objects

Managed Object Context = Object Graph

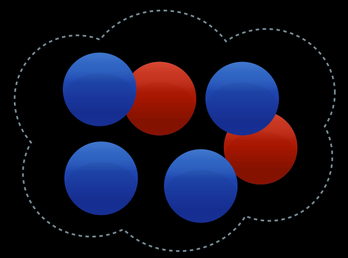
Model = Data Schema

Name	Ben
Name	Miguel
Name	Adam
Name	Melissa

Store = Data File

The Core Data Workflow

Introducing "The Stack"



Store File

Name Ben

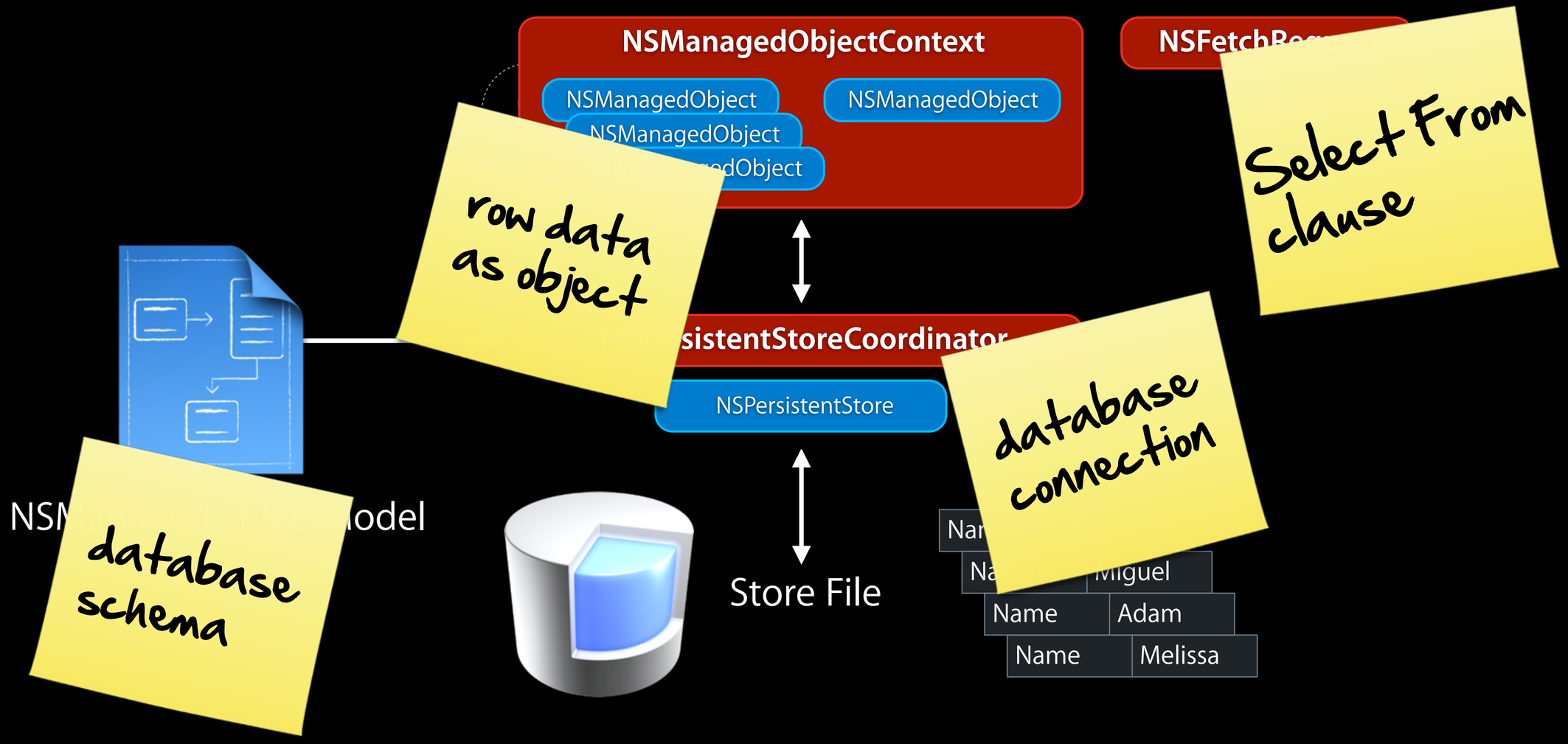
Name Miguel

Name Adam

Name Melissa

The Core Data Workflow

Introducing "The Stack"



Getting a Working Stack

In four easy steps

1. Load model
2. Create coordinator and set model
3. Add store
4. Create context and set coordinator

Demo

Xcode's Core Data template

Modeling Your Data

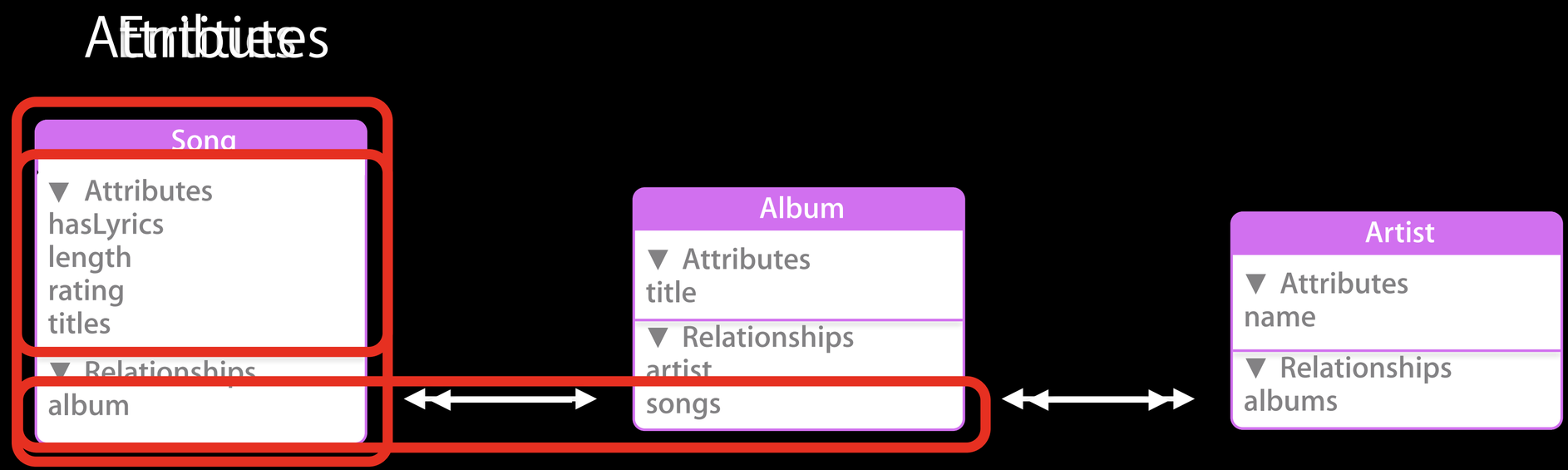
The Core Data Model

NSManagedObjectModel

- Structure of your data
  - Used to create store file
- Rules to maintain your data
  - Used to manage object graph
- Design it with usage patterns in mind

Model Building Blocks

An example



Relationships

Entities

NSEntityDescription

- Blueprints for managed objects
- Made up of properties
  - Attributes
  - Relationships
- Can have transient properties
  - Computed and non-persisted

Attributes

NSAttributeDescription

- Indicate value's type
  - Numeric (int, decimal, double, float, bool)
  - String
  - Date
  - Binary
  - Transformable
- Can have simple validation rules

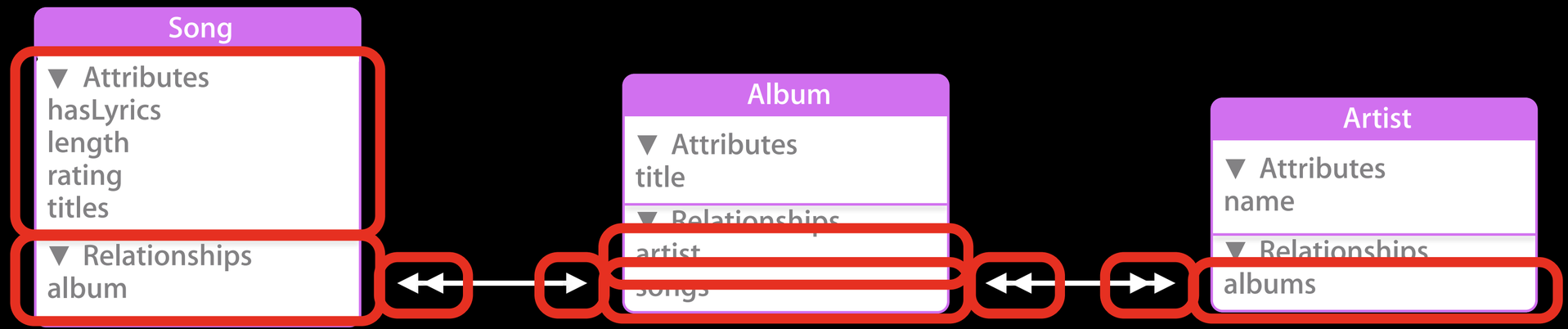
Relationships

NSRelationshipDescription

- Properties referencing other entities
- To-One
- To-Many
  - Unordered
- Bidirectional (inverses)
- Delete Rule

Music Data Model

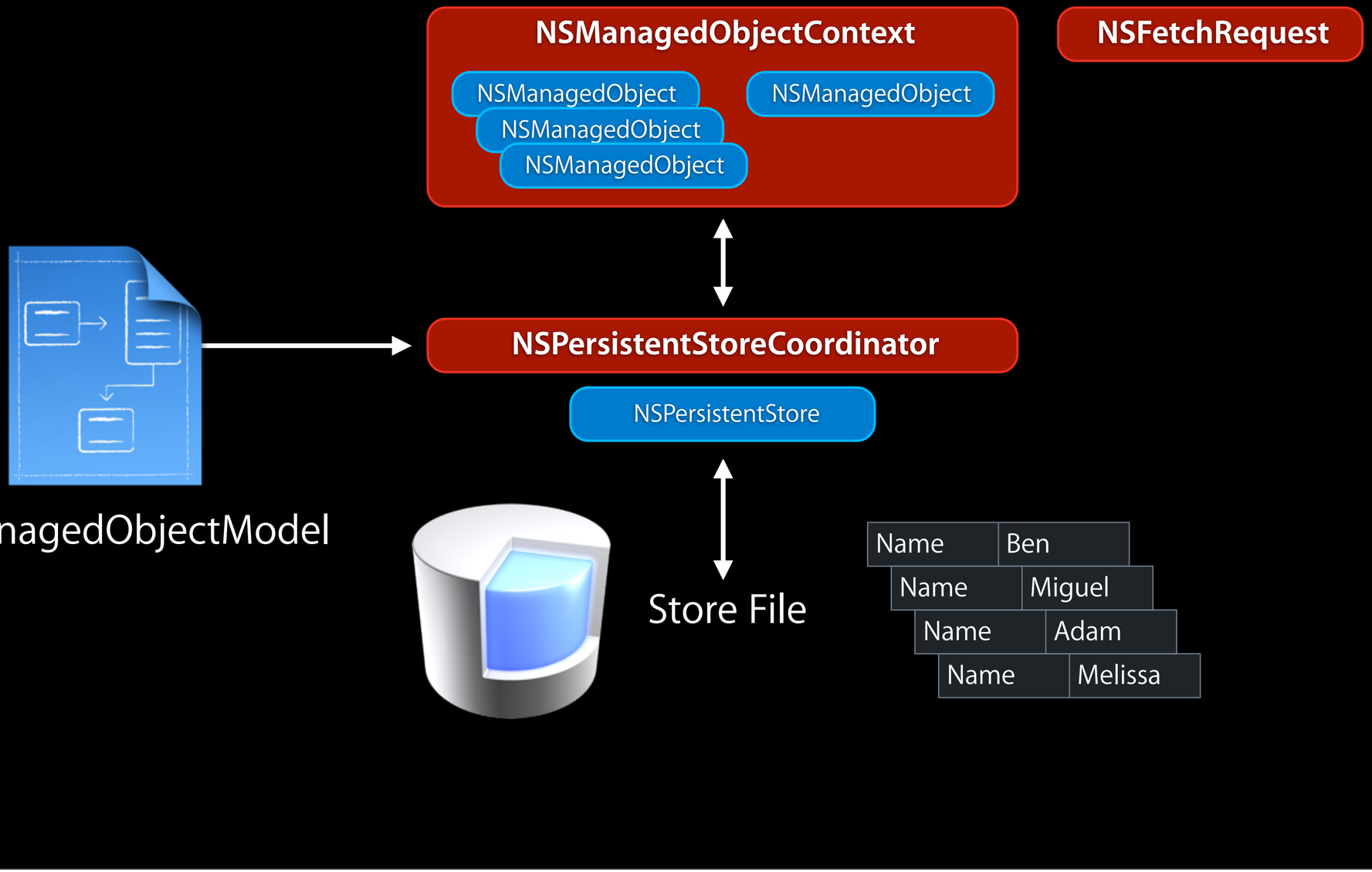
Revisited



Working With Managed Objects

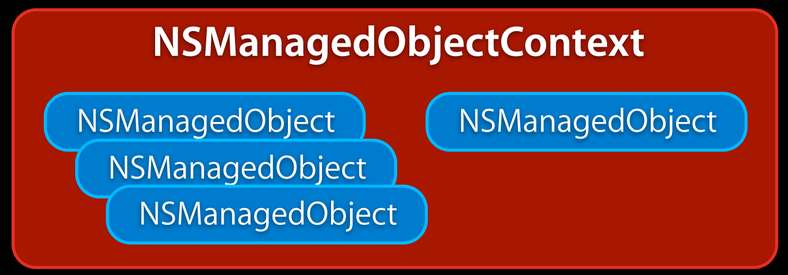
Where Are We?

“The Stack”



Where Are We?

“The Stack”



NSManagedObjectModel

Managed Objects

The data workhorse

- Instance represents a “row” of data
- Associated with single entity
- Has unique ID
  - Object wrapper for primary key

Interacting with Managed Objects

Tips

- Use accessors, like any other object
- Objective-C properties
- To-Many relationships are sets

NSManagedObject

Basic use requires no code generation

- Default implementation provides KVC/KVO support
- Extend behavior to
  - Compute derived value
  - Multi property validation

- Use Xcode
  - New File > NSManagedObject Subclass

The Workspace for Your Objects

NSManagedObjectContext

- Objects must be associated with a context
  - Weak reference
- Context observes object changes
- Represents a distinct set of edits

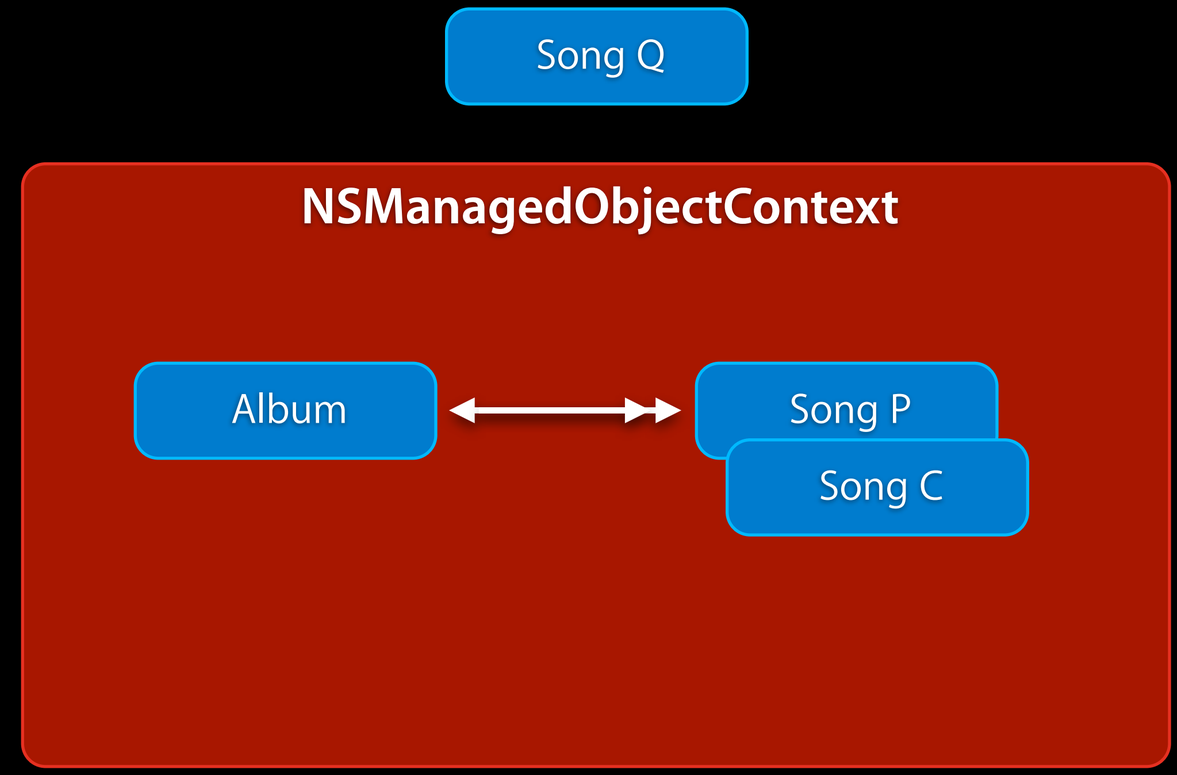
If Objects Are Nouns, Context Defines Verbs

NSManagedObjectContext actions

- Fetches
- Inserts
- Deletes
- Saves
- Tracks changes to modeled properties

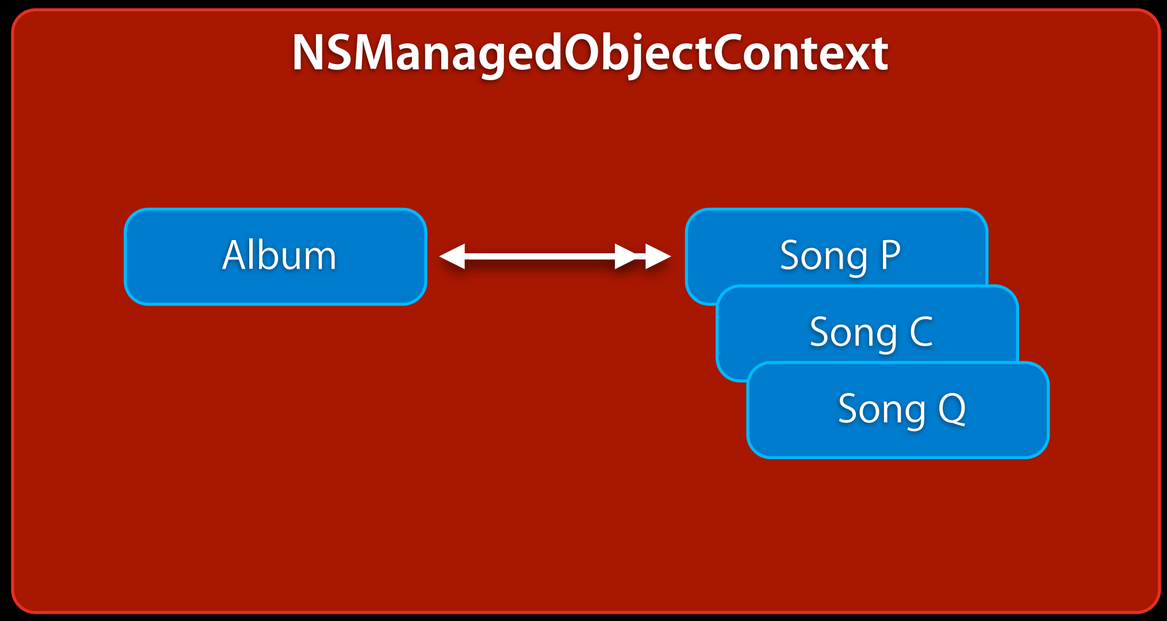
Example: Adding a Song

Adding to the object graph



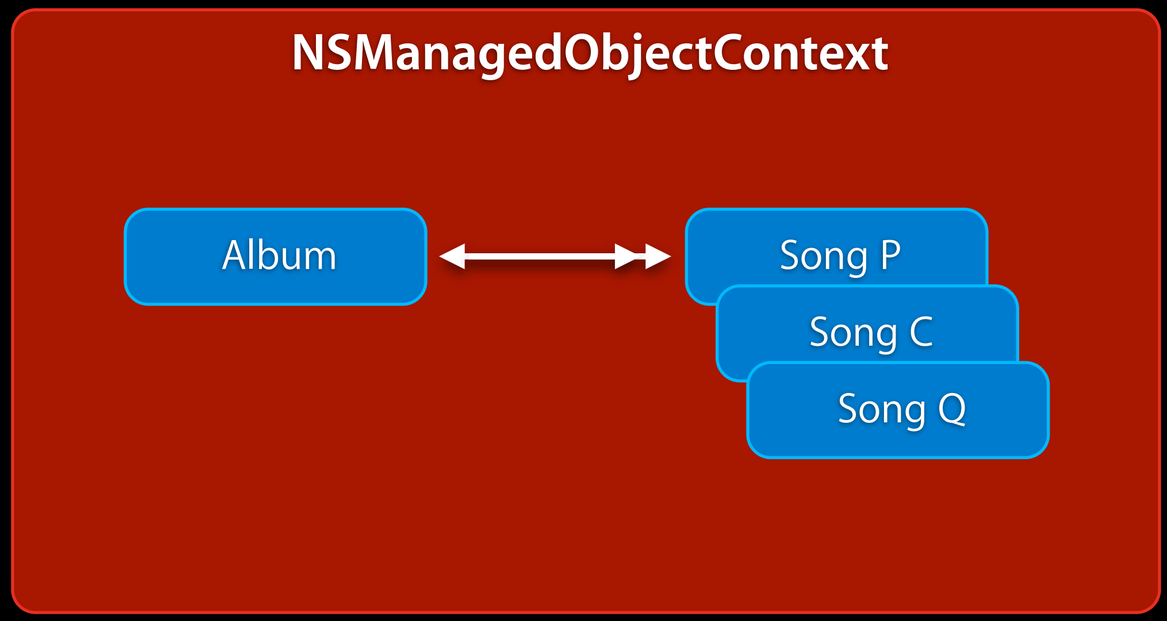
Example: Deleting Album

Cascading changes



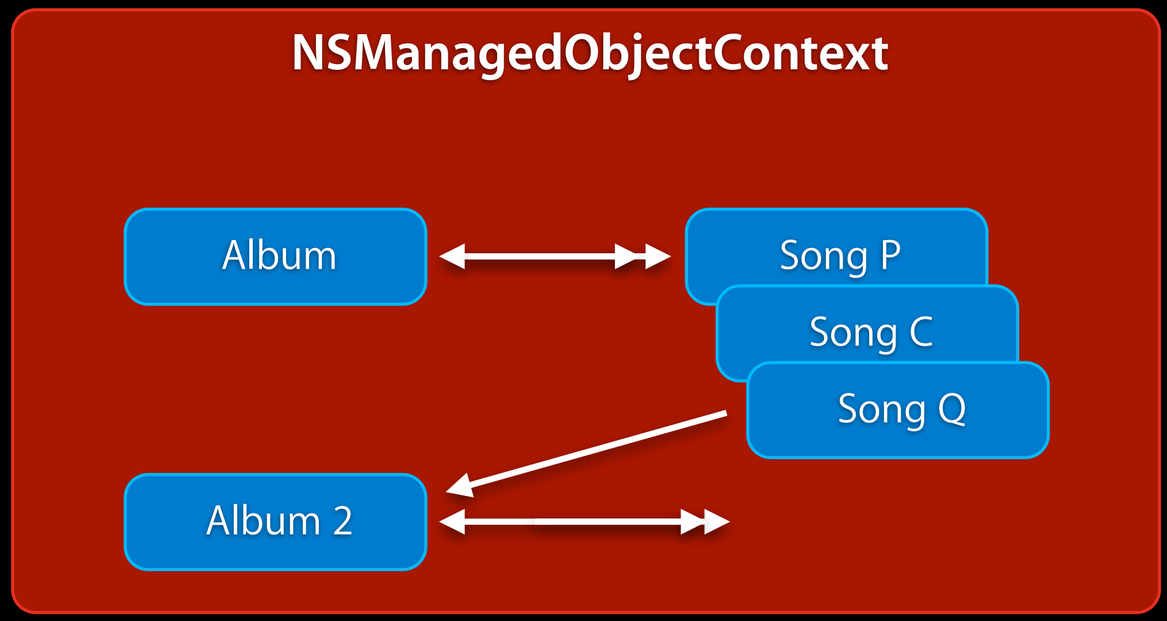
Example: Changing Your Mind

Undo



Example: Moving a Song

Relationship maintenance



Fetching

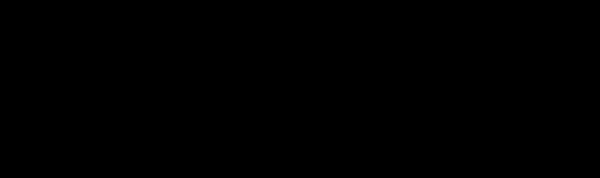
Where Are We?

Fetching

NSManagedObjectContext

NSFetchRequest

NSManagedObjectModel



Store File

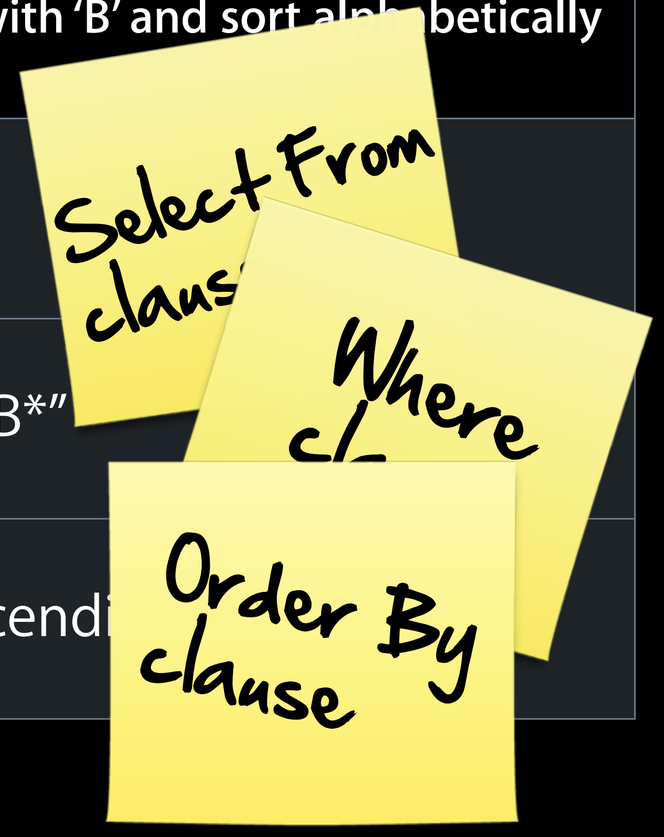
Name	Ben
Name	Miguel
Name	Adam
Name	Melissa

Describing Your Query

NSFetchRequest

- Entity
- Filter predicate
- Sort descriptors
- Optimization settings

Fetch all Albums beginning with ‘B’ and sort alphabetically	
Entity	ALBUM
Predicate	title like “B*”
Sorting	name: ascendi



Fetching All Artists

Example one

Fetch all Artists	
Entity	ARTIST
Predicate	
Sorting	name: ascending alphabetical

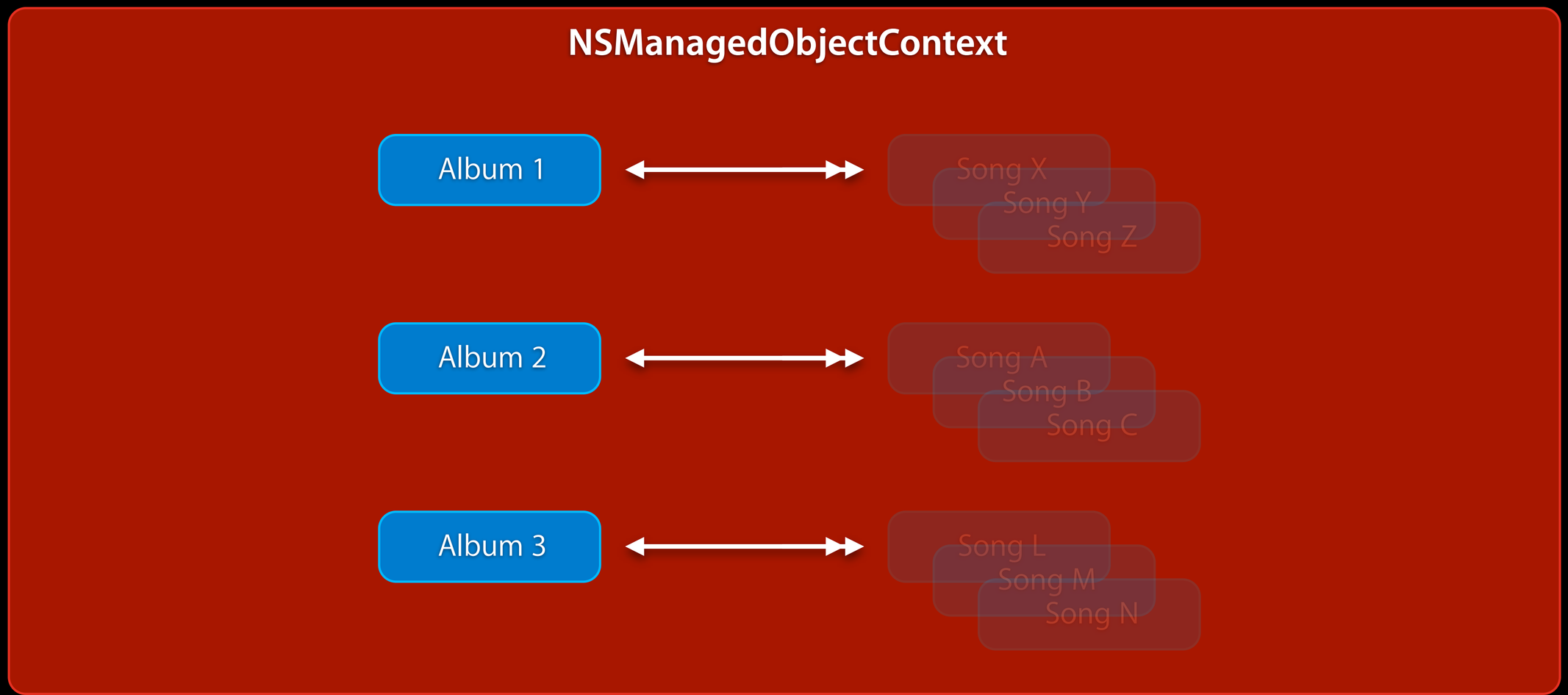
Fetching All Albums with at Least 10 Songs

Example two

Albums with more than 10 songs	
Entity	ALBUM
Predicate	songs.@count > 10
Sorting	name: ascending alphabetical

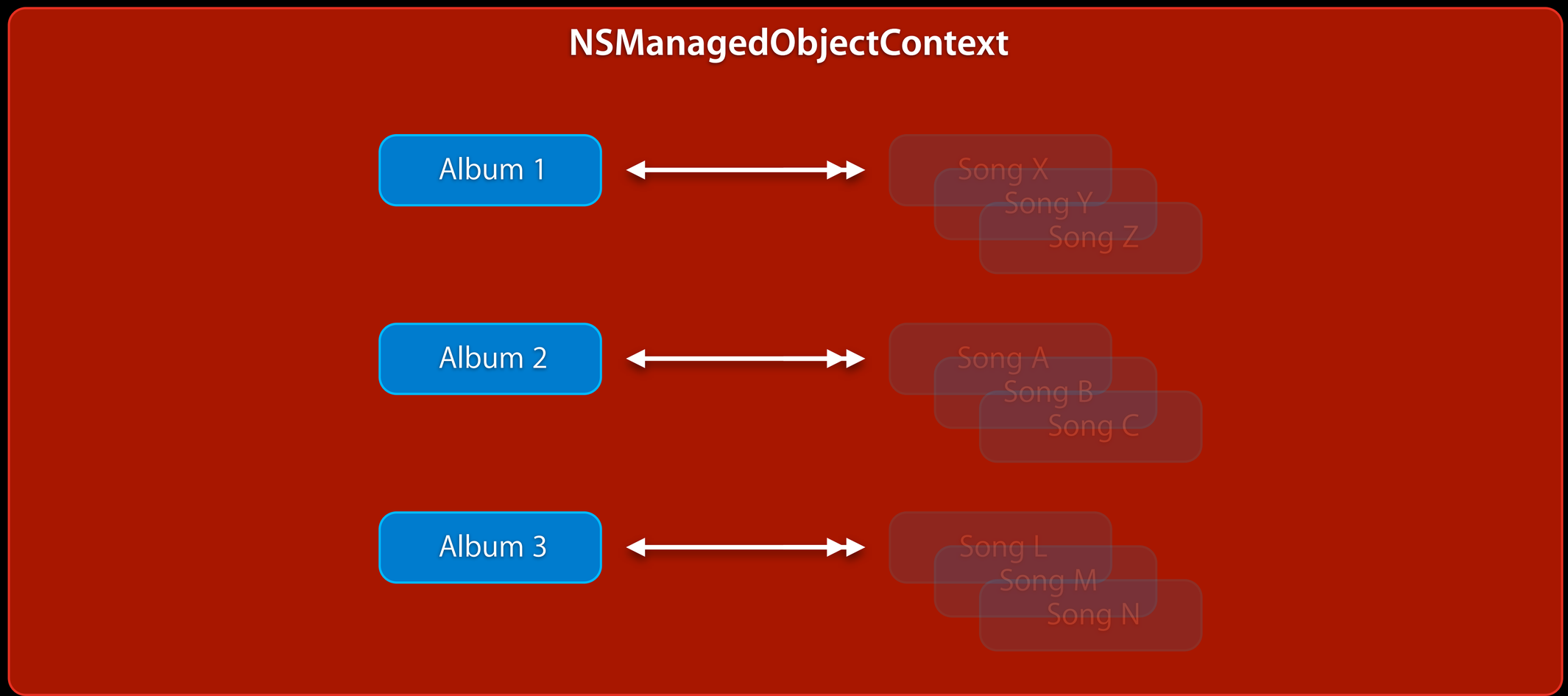
How Much Is Fetched?

Introducing "Object Faults"



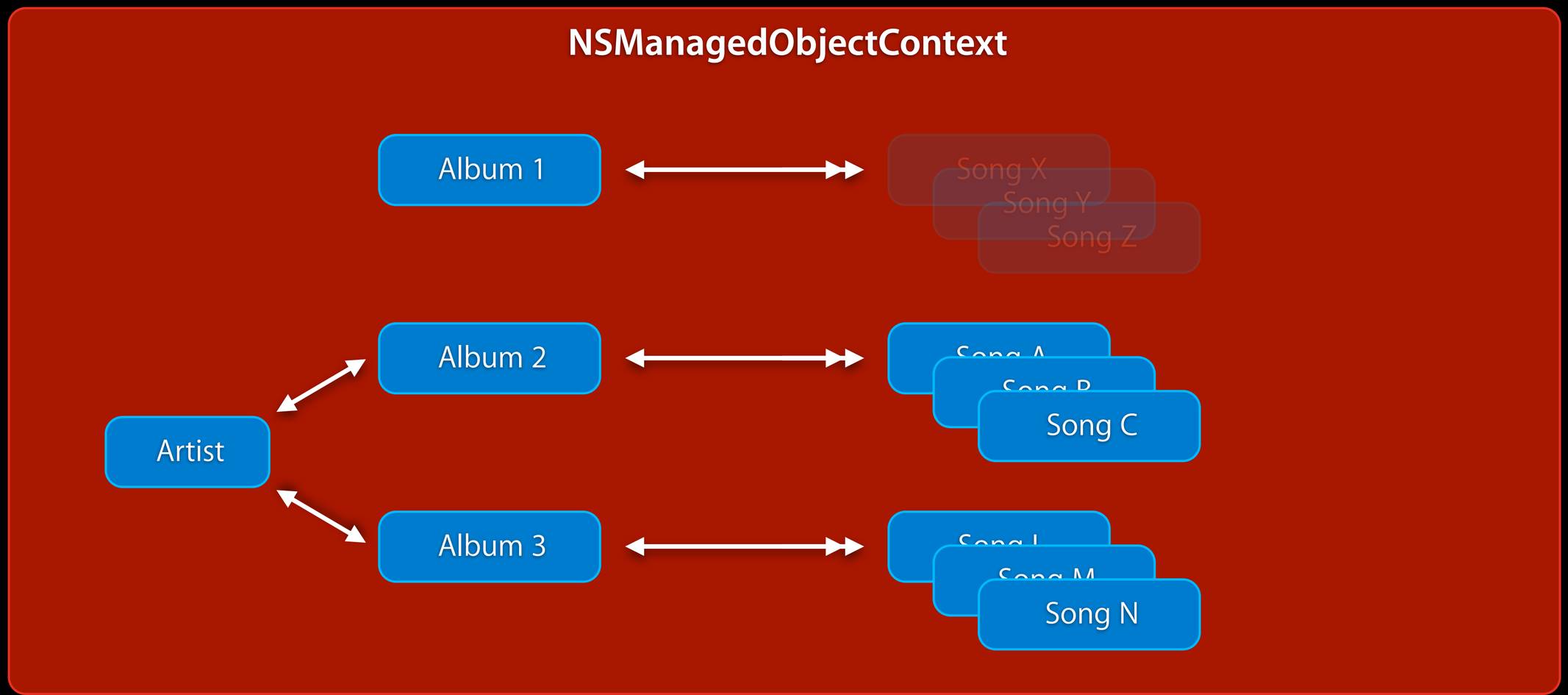
Traversing the Object Graph

Firing of faults



How Many Object Copies?

Single instance per context



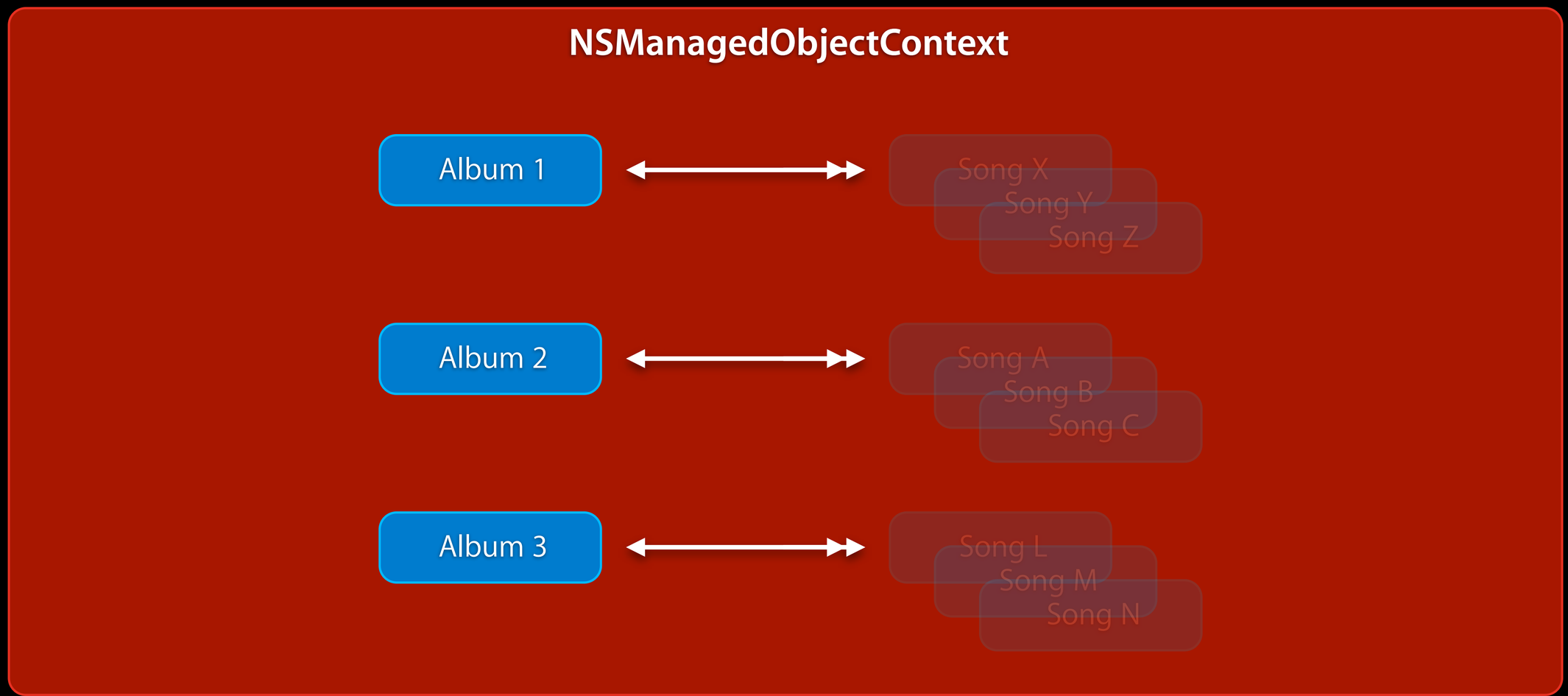
Improving Fetching Performance

How are you actually using the result?

- Just need a count
- Don’t need full objects
- Always require related objects

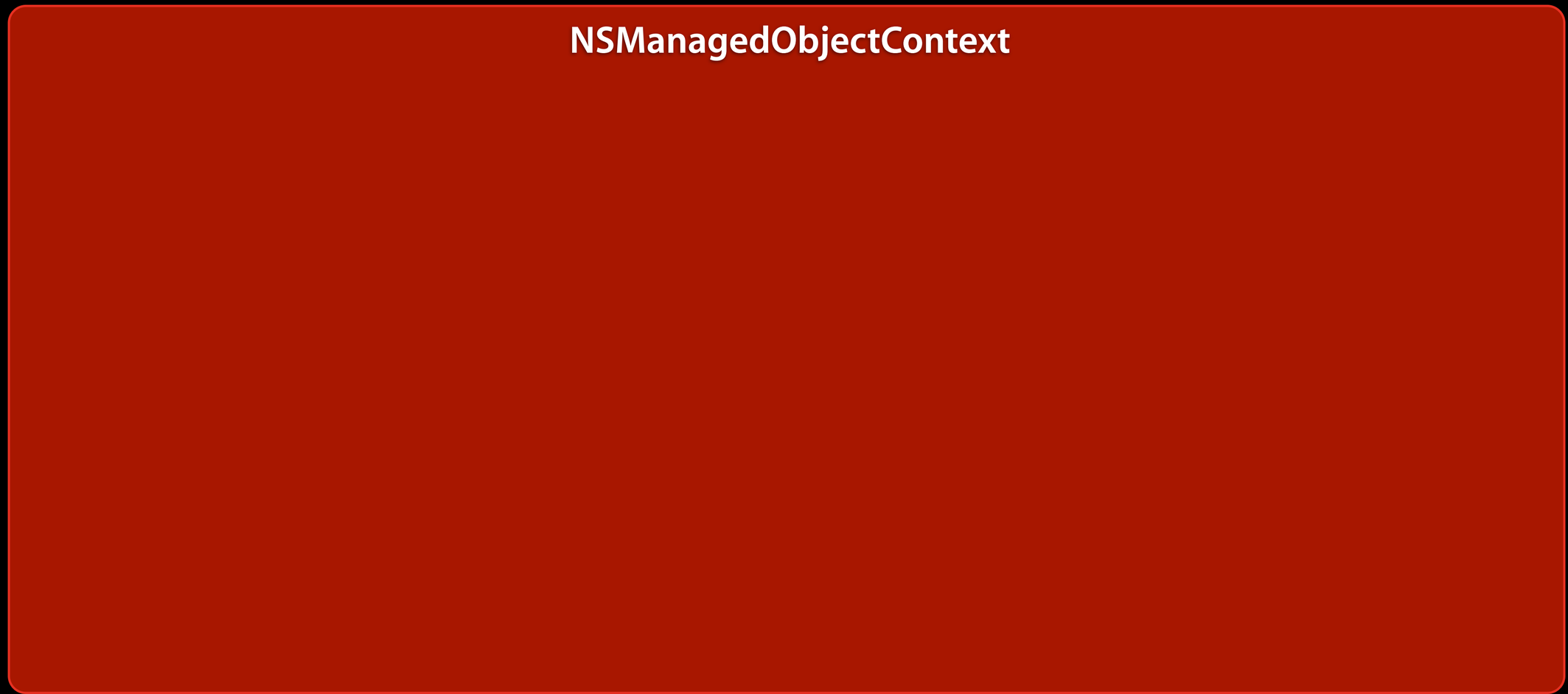
Typical Fetch

Fetching albums



Getting Only a Count

countForFetchRequest:

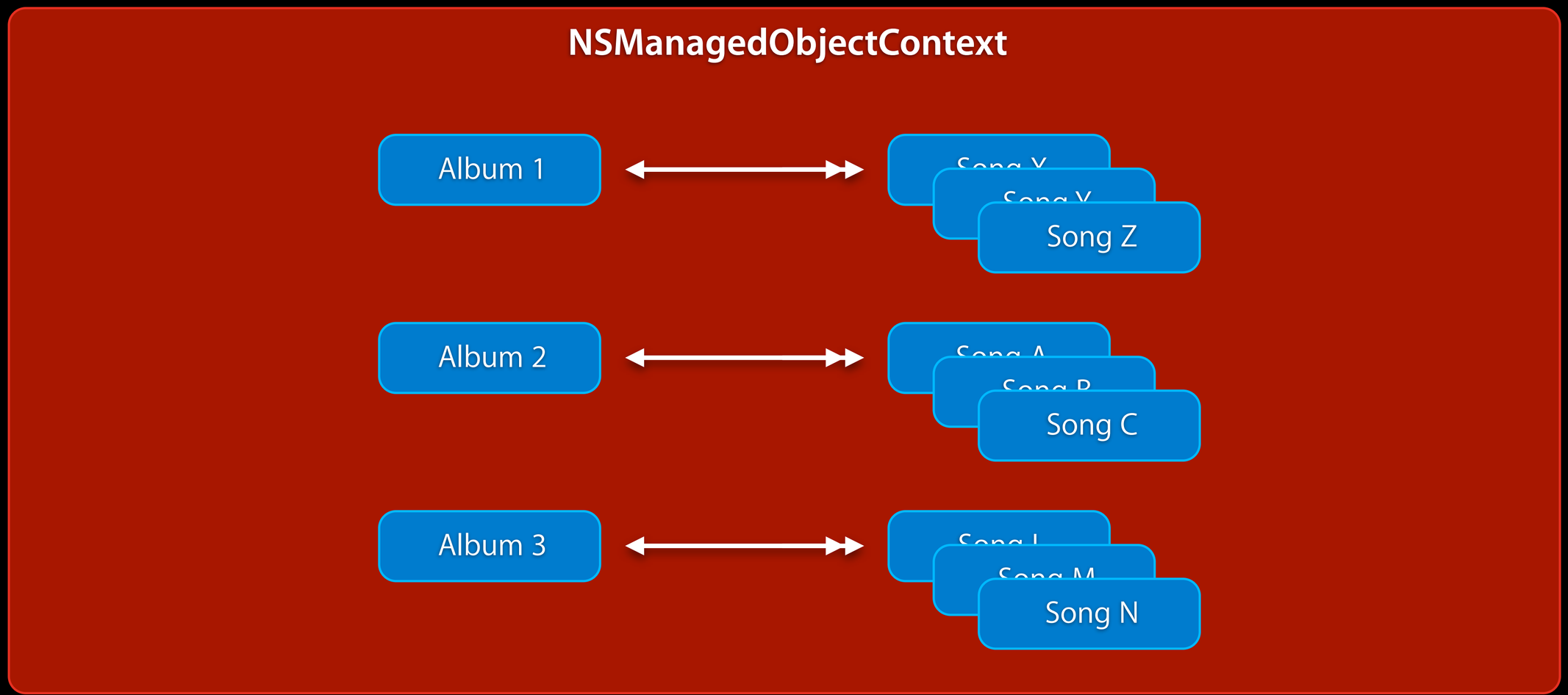


Fetching Subset of Properties

setPropertiesToFetch:

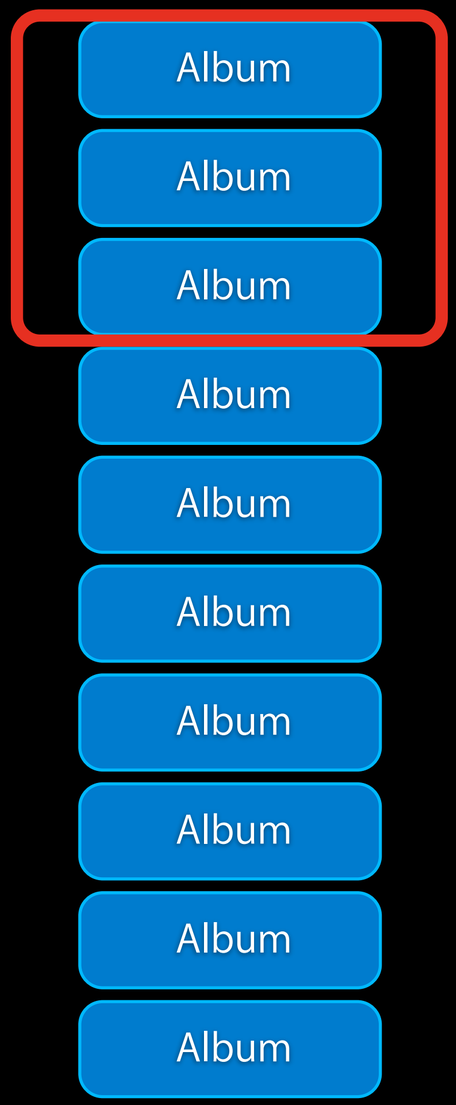
Prefetching Relationships

setRelationshipKeyPathsForPrefetching:



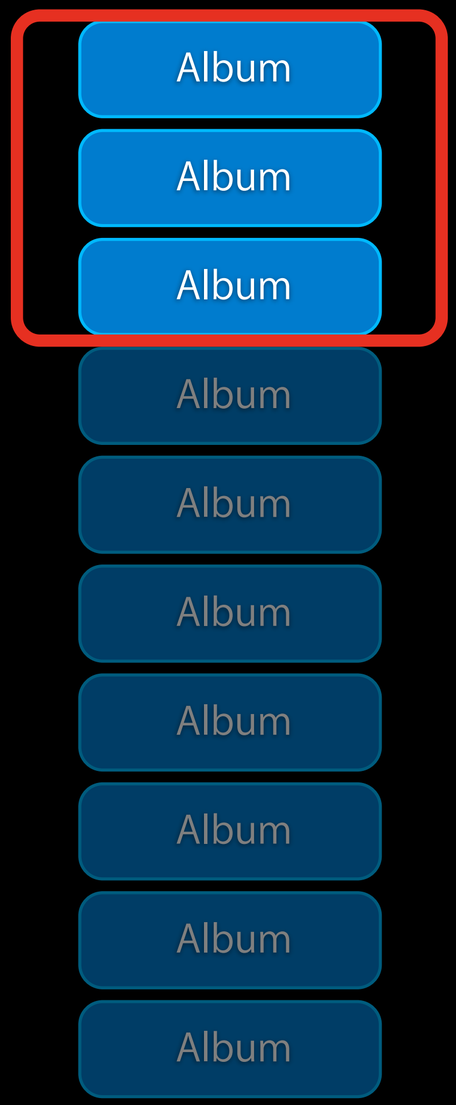
Fetching in Batches

You only need a group of objects at a time



Fetching in Batches

setBatchSize:



Customize Fetching to Usage Pattern

Study NSFetchRequest

One line of code =
Huge Performance Wins

Working with UIKit

Tips for Displaying Data

Core Data primarily helps with back end

- Managed Objects are just objects, use standard UIKit classes
- Leverage change tracking notifications
  - Register for object level changes (KVO)
  - Register for graph level changes (NSNotifications)

Displaying Data with Table Views

UITableView requirements

- List of data
- Divided into sections
  - How many sections?
  - How many objects per section?
- Core Data can help
  - NSFetchedResultsController

Breaking Up Fetches into Sections

NSFetchedResultsController

- Fetch request provides
  - Predicate to qualify fetch
  - Descriptors to sort results
- Add **keypath** returning section name
  - We compute and cache section information
  - We update information by observing changes

Performance Benefits

Caching section data

- Settings on NSFetchedResultsController are immutable
  - Fetch request
  - Predicate
- Create separate controllers for different data sets

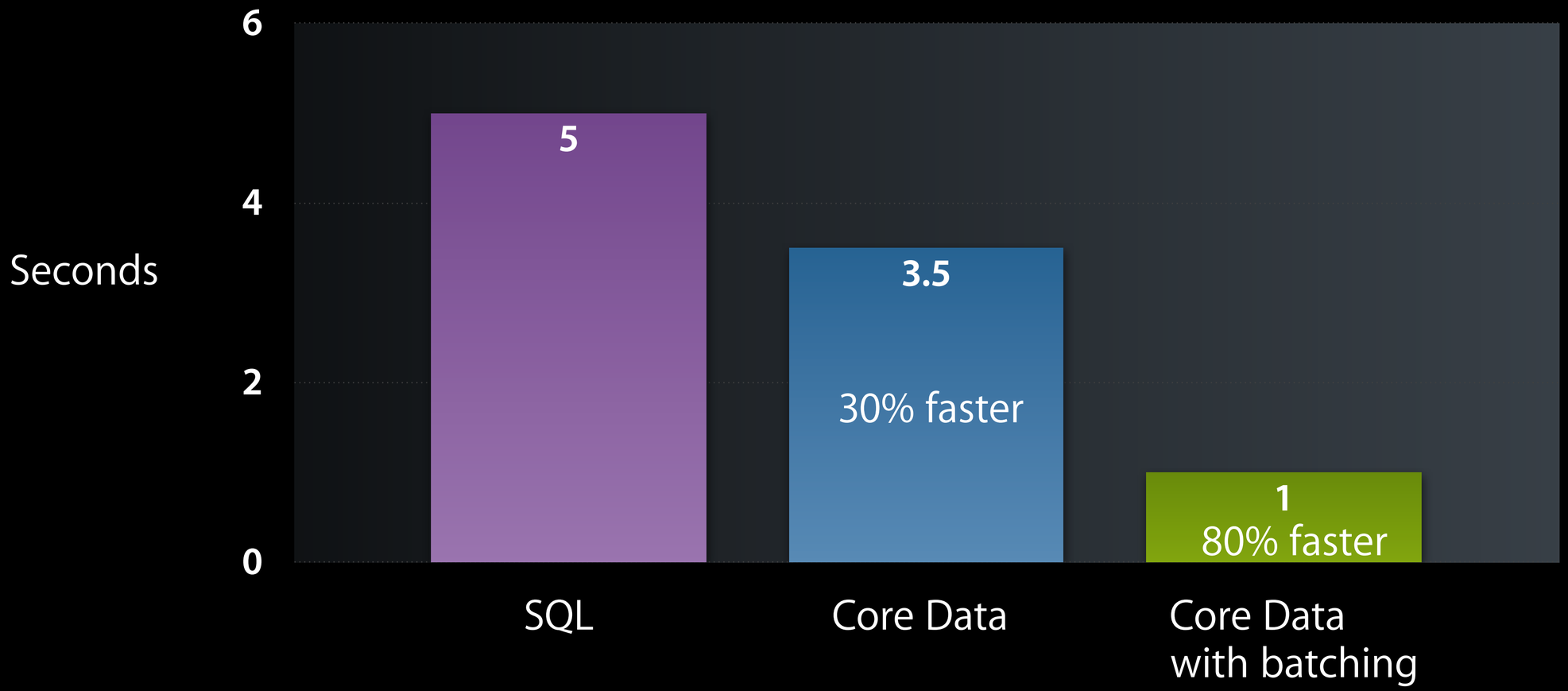
Demo: TopSongs

Using NSFetchedRequestsController

Performance

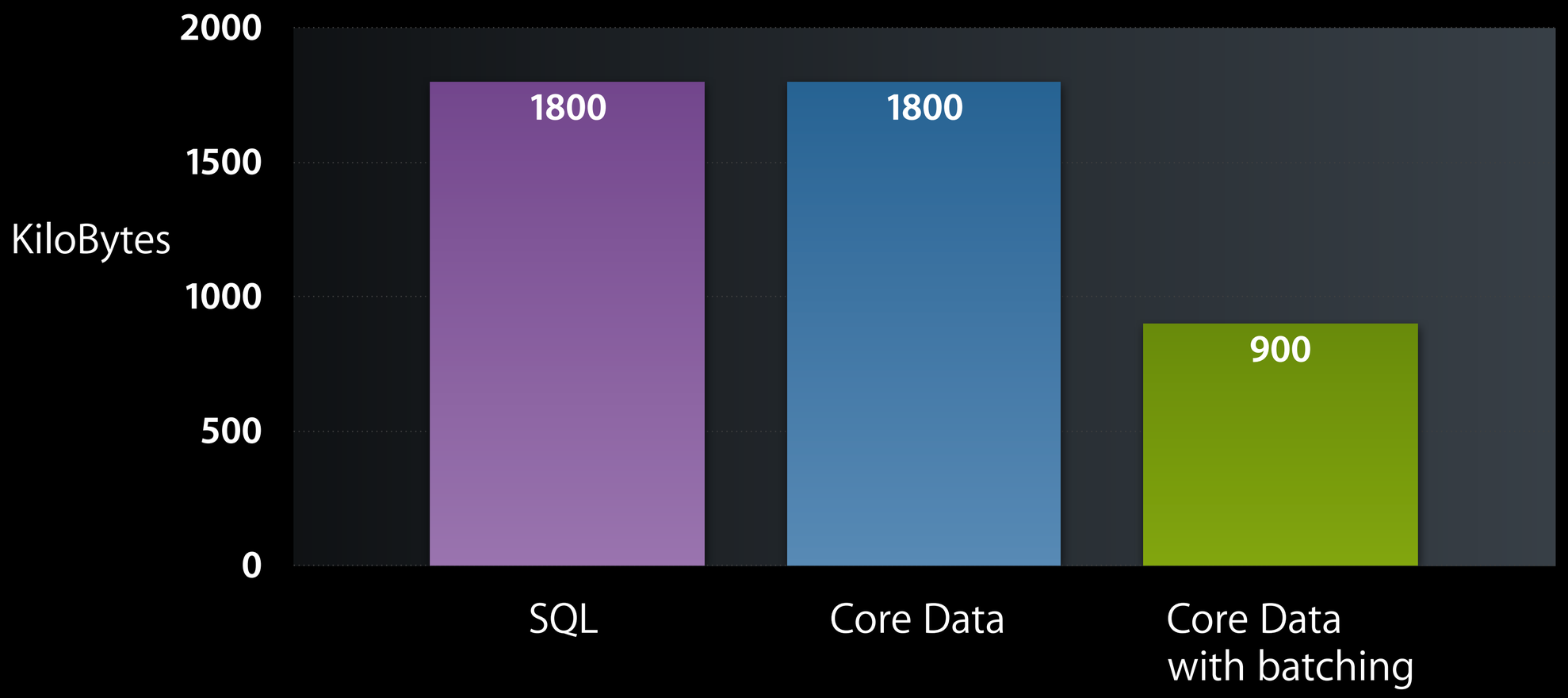
Speed on iPhone

Launch time 10K records of sample application



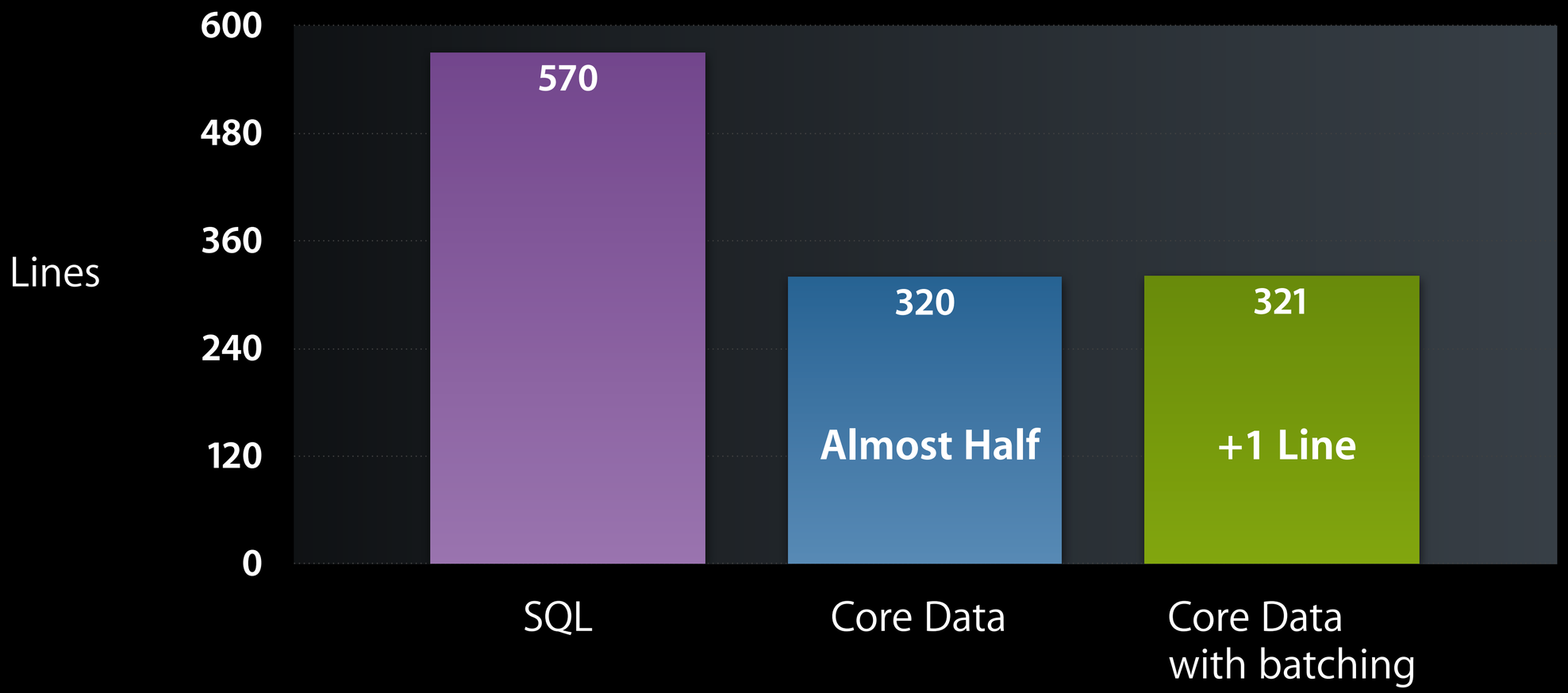
Memory Use on iPhone

Active heap size



Lines of Code

Less DB grunt work



Next Steps

Using Preexisting Files

Import

- Core Data owns store file for creation and maintenance
  - Don’t touch!
- One time import of data
  - Create managed objects
  - Save context

Documentation
Core Data Programming Guide

Evolving the Data Model

Breaking the contract

- Opening store with incompatible model results in error

```
if (![persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
                                                configuration:nil
                                                URL:storeUrl
                                                options:nil
                                                error:&error]) {

        // Handle error

        NSLog(@"ERROR: %@ : %@",error,[error userInfo]);
}
```

NSPersistentStoreIncompatibleVersionHashError

"The model used to open the store is incompatible with the one used to create the store"

Evolving the Data Model

What to do after a version error?

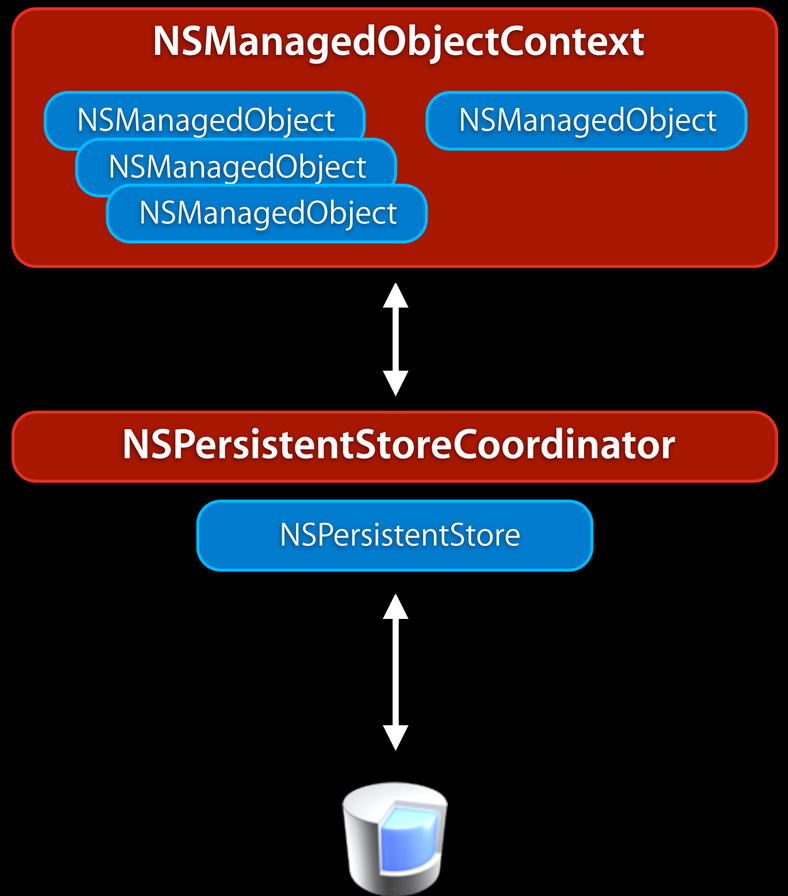
- If working with throwaway data
  - Delete store file and relaunch
- If need to migrate data
  - Use Core Data Mapping Model
  - Using lightweight migration

Documentation

Core Data Migration Guide

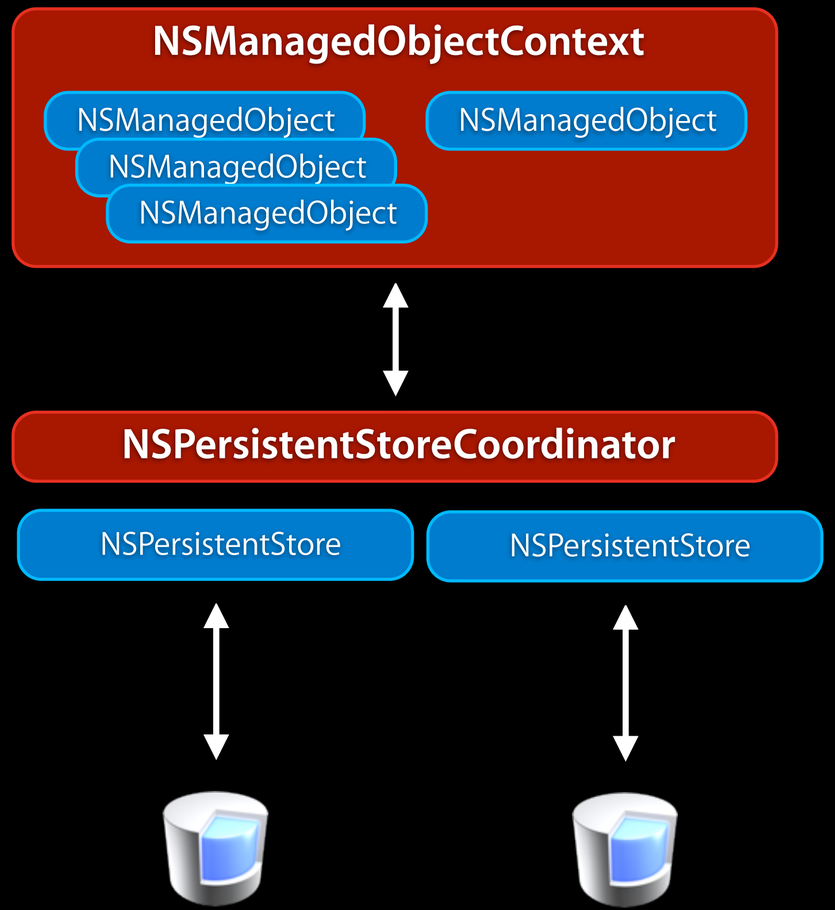
Advanced Stack Configurations

To fit your data and access patterns



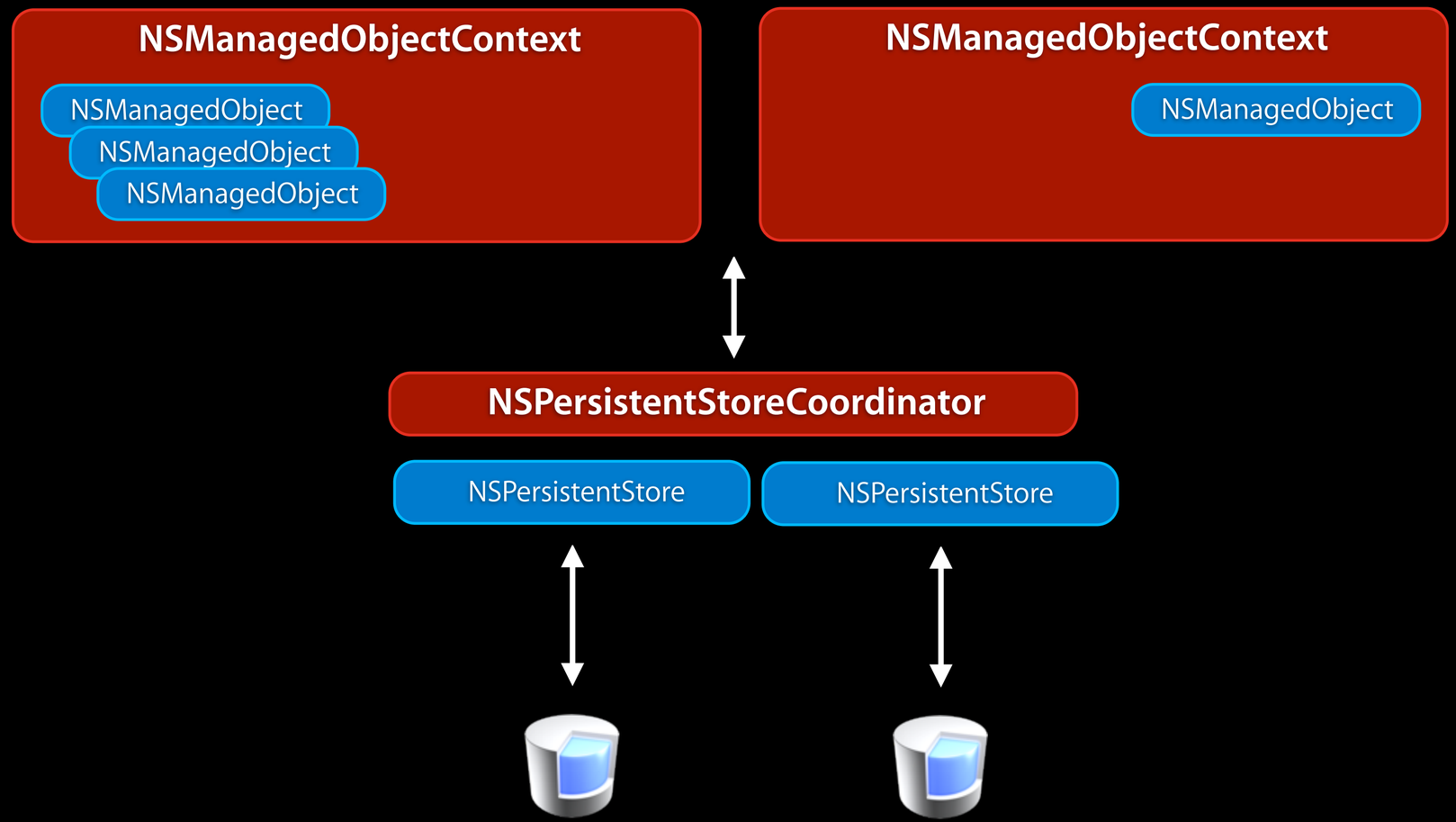
Advanced Stack Configurations

To fit your data and access patterns



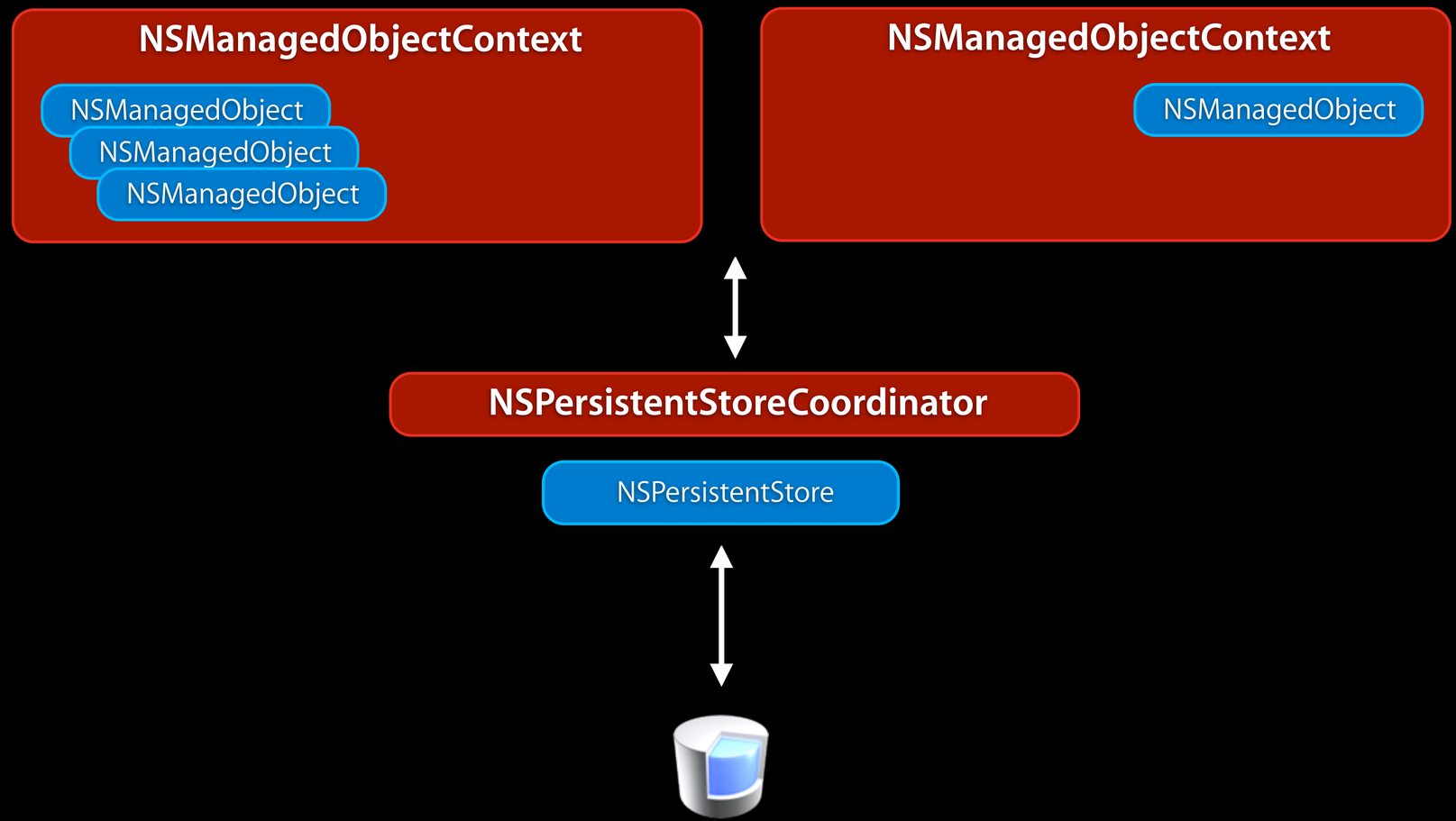
Advanced Stack Configurations

To fit your data and access patterns



Advanced Stack Configurations

To fit your data and access patterns



Core Data on Mac OS X

Mature code base (Third release)

- Store File Format is same
  - Desktop also includes XML store type
- Can use your same managed object classes
  - Desktop also includes Cocoa Bindings UI support

More Information

Michael Jurewitz
Developer Tools Evangelist
jurewitz@apple.com

Core Data Documentation
Programming guides, examples, and tutorials
http://developer.apple.com

Starting Point: Core Data Overview

Related Sessions

What's New In Core Data	North Beach Wednesday 10:30AM

Labs

Core Data on iPhone Lab	iPhone Lab D Wednesday 9:00AM
Core Data on iPhone Lab	iPhone Lab D Wednesday 2:00PM
Migrating from SQLite to Core Data Tutorial Lab	OS X Lab C Thursday 12:00PM
Core Data Lab	Snow Leopard Lab Friday 9:00AM

Summary

- Mature and highly efficient technology …now on the iPhone
- Focus on creating a good model
- Think objects, not databases

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09

WWDC09

WWDC09